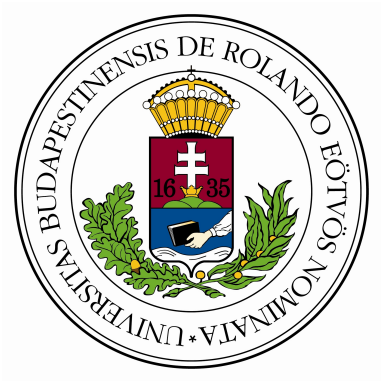
Dávid Ákos

Komponens-alapú szoftverek modellellenőrzéssel történő verifikációja

Tézisfüzet a Ph.D. értekezéshez

Témavezető: Dr. László Kozma C.Sc.



Eötvös Loránd Tudományegyetem

Az informatika alapjai és módszertana doktori program
Prof. János Demetrovics D.Sc.

Informatika Doktori Iskola
Prof. András Benczúr D.Sc.

Budapest
2012

1. Bevezetés

1.1. A komponens-alapú szoftverfejlesztés (CBSD)

Napjaink felgyorsult világában egyre kevésbé használhatók a vízesés-modellhez hasonló hagyományos szoftvertechnológiai módszerek. A sürgető határidők miatt a munkát részfeladatokra kell bontani, amelyen független csapatok dolgoznak [37]. Ez az igény hívta életre a komponens-alapú szoftverfejlesztést [21, 29, 31, 38, 42, 43, stb.]. A komponens-alapú szoftverfejlesztés vezérelvei az alábbiak [26]:

- Újrahasznosíthatóság, amely a már meglévő programrészek módosítás nélküli újbóli felhasználását jelenti.
- Fejleszthetőség, amely a komponens-alapú szoftverek karbantartási költségeinek alacsonyan tartását jelenti. Ez úgy lehetséges, hogy a rendszerkomponensek cseréje nincs hatással a rendszer többi részére.

A vezérelvek használatához az alábbi három feltétel megléte szükséges:

- Komponens-könyvtár
- Komponens-modell
- Szoftverarchitektúra

1.2. A modellellenőrzés alapfogalmai

A modellellenőrzés a szoftverrendszerek formális verifikációját algoritmikus szinten támogatja [16, 20, 35, stb.]. A modellellenőrzés segítségével teljes egészében automatizálható annak vizsgálata, hogy egy adott modell megfelel-e a formális specifikáció elvárásainak. A rendszer adott tulajdonságainak specifikációja általában temporális logikai formulák halmazának megadásával történik.
A modellt lehetőség szerint a forráskód speciális célnyelvre történő átírásával kell megadni. Az ilyen program megfeleltethető egy véges állapotgépnek (FSM), ami jellemzően egy csomópontokból és élekből álló irányított gráf. Minden egyes csomóponthoz atomi formulák egy halmaza társítható. A csomópontok jelképezik a rendszer egyes állapotait, az élek az állapotváltozást előidéző átmeneteket, míg az atomi formulák jelképezik a végrehajtás egy adott pontján fennálló alapvető tulajdonságokat. Bizonyos esetekben a modellellenőrzéssel történő verifikáció nem véges állapotú rendszerekre is alkalmazható, amennyiben különböző absztrakciós és indukciós szabályokkal kombináljuk.
Az állapottér robbanása okozza itt is a legnagyobb gondot, amely a sok – párhuzamos állapotátmenetet lehetővé tévő – komponensből álló rendszereknél jelentkezik, de az utóbbi időben komoly előrelépés történt az alábbi módszerek használatával: szimbolikus algoritmusok [13, 33, stb.], részleges sorrendiségen alapuló redukció [25, 36, 41,

stb.], absztrakció [15, 19, 22, 23, 30, stb.], szimmetria [24, 27, 28, 34, stb.], Bounded Model Checking (BMC) [17], valamint a különösen ígéretes – a rendszerek bővítésével csak a megváltozott részek újraellenőrzését megkívánó – nyílt inkrementális modellellenőrzés (OIMC) [39, 40, stb.]. A modellellenőrző eszközök általában saját specifikációs nyelvet használnak, de a legtöbbjük támogatja a lineáris temporális logika (LTL), illetve a kiszámítási fa logika (CTL) temporális logikai nyelveket.

2. A dolgozat felépítése

A 2. fejezetben formális módszerek segítségével vizsgáljuk a szoftverrendszerek komponenseinek granularitását. Rámutatunk arra, hogy a granularitás az állapottér, komoly, koncepcionális hibákat is elfedhet. A 3. fejezetben bemutatunk egy új módszert, melynek segítségével a sok hasonló komponensből álló rendszerek struktúrájában rejlő lehetőségek kiaknázásával csökkenthető az állapottér mérete. A 4. fejezetben a komponens-alapú rendszerek bővíthetőségének lehetőségeit vizsgáljuk. Bemutatunk egy új, általunk fejlesztett eszközt, amely támogatja az OIMC algoritmus használatát, valamint összehasonlítjuk az OIMC-t a saját módszerünkkel. Az 5. fejezetben új szempontból közelítjük meg a hálózati protokollok formális verifikációját, kiaknázva a komponens-alapú rendszerek, valamint a hálózatok rétegelt szerkezete közötti hasonlóságokat. A 6. fejezetben áttekintjük a témához kapcsolódó jelenlegi kutatási irányokat, kiemelve az esetleges kapcsolódási pontokat. A 7. fejezet az elvégzett kutatómunka összegzését tartalmazza.

3. Az elért eredmények

3.1. A komponensek granularitásáról

Roppant fontos még a tervezés korai szakaszában eldönteni, hogy az általunk készítendő komponens milyen szemcsézettségű legyen. Egy túlságosan finom komponens valószínűleg nem túl általános, így az újrahasznosíthatóság elve kerül veszélybe. Ugyanakkor egy túlságosan durva szemcsézettségű komponens jóval összetettebb interfészekkel rendelkezik, és sokkal valószínűbb, hogy szükség lesz a menet közbeni megváltoztatására. Ezért alapvető fontosságú az egyensúly megtalálása a komponensek méretének, illetve a rendszer összetettségének függvényében [42].
A 2.2 fejezetben a G. R. Andrews által javasolt szintézis módszer segítségével példákat hoztunk létre manuálisan [14]. A 2.3 fejezetben szintézis módszer segítségével a termelő-fogyasztó probléma specifikációját hoztuk létre, majd elkészítettük ennek általános és jól skálázható NuSMV modelljét, amellyel igazoltuk a rendszer megfelelő működését. Az általunk eredetileg közölt modellt felül kellett vizsgálnunk, miután felfedeztük, hogy bizonyos megszorítások elfedték a kiéheztetés egyik speciális esetét.

1. Tézis. *Formális eszközök segítségével rámutattunk arra, hogy absztrakció körültekintő alkalmazásával csökkenthető az összetett rendszerek esetében jelentkező ál-*

lapotrobbanás kockázata, ugyanakkor a komponensek nem megfelelő szemcsézettsége koncepcionális szinten rejtve maradó hibákat eredményezhet.

3.2. A sok hasonló komponensből álló rendszerek modellellenőrzését javító egyszerű módszer bemutatása

Egy rendszer sok hasonló komponensből történő létrehozása segíthet az állapottér csökkentésében, mivel a komponensek lokális specifikációjában szereplő formulahalmaz kiértékelését feleslegessé teszi. Erre mutattunk egy valós jellegű példát a 3.3 fejezetben. Észrevettük, hogy a komponensek közötti szerkezeti hasonlóságból fakadó előny kihasználható, hogy az egyes komponensek lokális specifikációjában szereplő formulák számát csökkentsük.
A megközelítésmód újdonsága abban rejlik, hogy az ilyen rendszerek formális modelljénél megadott specifikációs formulákat átstrukturáltuk aszerint, hogy kizárólag az adott komponensen belüli állapotokra hivatkozik-e. A feltételnek eleget tevő formulákat ideiglenesen kivettük a modellellenőrzés hatálya alól, és csak egyetlen alkalommal értékeltük ki. Ezzel tulajdonképpen a modellellenőrzést az osztályok szintjére emeltük, időt és erőforrást takarítva meg. A módszer egyszerű példákon történő bemutatása, valamint a kísérleti eredmények a 3.3.1 és a 3.3.2 fejezetekben találhatók.
A módszerünk redukciós szabályainak alkalmazásával a modellellenőrző eszköz kényszeríthető, hogy hagyja figyelmen kívül az érintett formulákat. Ez alapján az alábbi eredményre jutottunk: az ellenőrzött formulák száma, az állapottér mérete, valamint az igénybe vett erőforrások (idő, memória) mértéke érzékelhetően csökkent. A csomópontok számában történő százalékos csökkenés mértéke 2, 3, valamint 4 repülőgép esetében 34%; 26%; 23% sorrendben. A minta túlságosan kicsi, hogy további következtetéseket vonjunk le belőle, de az állapottér csökkenésének tényét jól mutatja.

2. Tézis. *Az általunk javasolt módszer alapvető lépései az alábbiak:*

1. *A rendszer megoldását megadjuk a legegyszerűbb esetre. Például, a megoldást egyetlen komponensre hozzuk létre.*

2. *Végrehajtjuk a rendszer modellellenőrzését. Amennyiben a folyamat sikeres, továbblépünk a 3. lépésre. Ha a modellellenőrző eszköz hibával tér vissza, a megoldás felülvizsgálata és szükség szerinti módosítása esedékes. A modellellenőrzést addig ismételjük, amíg az elvárt eredményeket meg nem kapjuk.*

3. *A megoldást új szempontok szerint elemezzük. Konkrétan azt nézzük meg, hogy a specifikáció szétbontható-e lokális és globális részre.*

4. *A rendszermodellt megadjuk több komponensre is. A globális és lokális specifikációt szétválasztjuk. Utóbbiak kimaradnak az ellenőrzésből, mivel a modellellenőrzést kizárólag a globális specifikációra végezzük el. Amennyiben a kívánt*

eredményeket kapjuk, a teljes rendszer helyesnek tekinthető mind a globális, mind pedig a lokális specifikációra nézve. Ellenkező esetben a rendszer felülvizsgálata és szükség szerinti módosítása esedékes.

3.3. Az OIMC összehasonlítása a saját módszerünkkel

Nem könnyű a két módszer összehasonlítása, mivel jelenleg egyik sincs a gyakorlatban implementálva. Bár támaszkodhatunk a kiszámított csomópontok számára, úgy döntöttünk, hogy más szempontokat veszünk figyelembe: *a számítási költség, teljesítmény, valamint az oktatási szempont.*
Egyrészt kísérleti adatokkal igyekeztünk alátámasztani az OIMC módszer gyakorlati alkalmazhatóságát és hatékonyságát, melynek eredményeként körülbelül 78%-kal csökkent a 4.5.3 fejezetben bemutatott repülőtéri példa állapottere.
Másrészt rámutattunk a módszer jelenlegi korlátaira, mint amilyen például a valós rendszerek modelljeinél viszonylag gyakran előforduló körkörös függőség esete, amelynek megfelelő kezeléséhez a legnagyobb fixponttal történő közelítést javasoljuk, de ezen jelen pillanatban is dolgozunk.
Arra is rámutattunk a 4.5.2 fejezetben, hogy milyen fontos az elégséges feltételek megfelelő meghatározása egy minimálisan biztonságos rendszer eléréséhez. A jelenség rávilágít a rendszermodellezésnél adódó problémára, mely szerint a nem elégséges specifikáció megadásával vagy hibás formulák használatával olyan nem biztonságos rendszerek hozhatók létre, amelyek a formális módszerek legfőbb előnyét ássák alá, nevezetesen a garanciát, hogy nem marad rejtett hiba a modellben vagy a kódban.

3. Tézis. *A 3.3 és 4.5 fejezetekben szereplő kísérleti adatokra alapozva azt gyanítjuk, hogy nincs jelentős különbség a fenti két algoritmus között abban az esetben, ha a rendszermodell hasonló vagy azonos komponensekből áll.*

Ennek oka egészen egyszerű; míg az OIMC bonyolultabb algoritmust használ a formulák lezártjainak előállításához és összehasonlításához, addig az általunk bemutatott módszer több előkészítő munkát igényel a formulák elemzése, illetve átstrukturálása során. Azt viszont fontos kiemelni, hogy az OIMC sokkal általánosabban használható megoldás, mivel nem támaszt előfeltételeket a komponensekkel szemben.

3.4. Az OIMC használatát támogató JanKo eszköz bemutatása

Mivel az OIMC algoritmus jelenleg nincs implementálva, ezért kifejlesztettünk egy JanKo nevű egyszerű Java alkalmazást, amely támogatja az inkrementális modellellenőrzés használatát. Habár az eszköz teljesen működőképes, továbbra is egy fejlesztés alatt álló munkának tekintendő. A végső cél az eszköz funkcionalitásának az NuSMV modellellenőrző eszközbe történő teljes integrációja.

A JanKo eszköz automatikusan kinyeri és megjeleníti a bemeneti fájlokban lévő érvényes formulákat. Egyaránt felismeri a CTL, illetve az LTL formulákat, de egyszerre csak egyik típussal tud dolgozni (a középső panel rádiógombjával szabályozható). Fontos kiemelni, hogy bár az eredeti OIMC algoritmus csak a CTL temporális logikai nyelvet támogatja, az általunk kifejlesztett eszköz az LTL formulákat is kezeli.
Habár két (bemeneti fájlok, illetve manuálisan bevitt formulahalmazok által reprezentált) komponens közötti kompatibilitás eldöntéséhez továbbra is szükség van felhasználói beavatkozásra, az eszköz már így is komoly előrelépés az OIMC gyakorlatban történő automatizálásához.
Jelenleg is dolgozunk a JanKo NuSMV modellellenőrző eszközbe történő integrációján. Első lépésként folyamatban van az eszköz felkészítése a bemeneti fájlok, mint parancssori paraméterek támogatására. Továbbá felvettük a kapcsolatot a Roberto Cavada és társai neve által fémjelzett kutató- és fejlesztőcsoport tagjaival, hogy közös kutatást kezdeményezzünk a témában [18].

4. Tézis. *Az általunk fejlesztett eszköz feladata, hogy a báziskomponens és a bővítmény fájlba történő ellenőrzési eredményeit bemenetként használva automatikusan elvégezze az algoritmus első lépéseit, kinyerje a specifikációban szereplő formulákat, majd előállítsa ezek lezártjait. Az eszköz végül lehetőséget kínál a két komponens formulahalmazaiból előállított lezártjainak vizuális összehasonlítására. Az eszköz forráskódja az A függelékben található.*

3.5. Hálózati protokollok formális verifikációja

A hálózati protokollok a nagyméretű és komplex szoftverrendszerek sajátosságait hordozzák, ezért egymásra épülő, hierarchikus rétegekbe szervezik őket (például az ISO/OSI referenciamodell szerint). A rétegelt szerkezet lehetővé teszi, hogy az egyes kommunikációs funkciókat egymástól függetlenül definiáljuk, ezzel együtt ezek fejlesztése is külön – eltérő szabványok szerint – történjen. Egy adott réteghez tartozó protokoll egyfajta fekete dobozként képzelhető el, amely úgy kínál szolgáltatásokat a felette lévő rétegnek, hogy a vele egy szinten lévő egyedekkel kommunikálva igénybe veszi az alatta lévő réteg szolgáltatásait. A tényleges protokoll az adott réteg kommunikációs szabályaiból és a felette lévő rétegnek nyújtott szolgáltatásokból (szolgáltatásspecifikáció) tevődik össze. A protokoll egyedeinek leírását, valamint a szolgáltatásspecifikációt együttesen protokollspecifikációnak nevezzük.

5. Tézis. *Új megközelítésből tárgyaltuk a hálózati protokollok ellenőrzési lehetőségeit. Az általunk bemutatott módszer szerint az OSI/ISO rétegmodelljét komponens-alapú rendszernek tekinthetjük, mivel az egyes rétegek fekete dobozként működnek, és kizárólag az interfészeiken keresztül kommunikálnak egymással. Minden egyes réteg kizárólag az alatta lévő réteg szolgáltatásait veheti igénybe, és kizárólag a felette lévő rétegnek nyújthat szolgáltatást.*

Példaként az Open Shortest Path First (OSPF) protokollt vettük, amely egy nagy hálózatokban is jól használható belső irányítóprotokoll. Sikeresen létrehoztunk egy

olyan modellt, amely a hivatalosan elérhető adatokon alapul, és igazolja egy jól működő OSPF irányítóprotokoll meglétét. A protokoll verifikációját illusztráló, az 5.3 fejezetben bemutatott állítások alátámasztották a feltevésünket, és a kívánt eredményt adták.

6. Tézis. *Erre az analógiára építve egy konkrét, széles körben használt irányítóprotokoll, az OSPF modelljét hoztuk létre a később szabvánnyá váló RFC dokumentum alapján. Formális eszközökkel igazoltuk, hogy a protokoll minden tekintetben az eredeti szolgáltatásspecifikációnak megfelelően működik.*

Az elkészült modell a későbbiekben tovább bővíthető újabb útválasztó eszközök bekapcsolásával. Így további hitelesítési funkciók, illetve a kapcsolatállapoti hirdetmények (LSA) szinkronizációs folyamatának vizsgálata és igazolása válik lehetővé. Az irányítóprotokoll vizsgálata során az egyik útválasztó családon futó Internetwork Operating System (IOS) operációs rendszerben egy komoly biztonsági rést fedeztünk fel. Ennek formális eszközökkel történő modellezése, illetve vizsgálata még nem zárult le. A biztonsági probléma részletei az 5.4 fejezetben találhatók. Sajnos, idő hiányában nem volt lehetőségünk sem a 2800-as sorozat többi típusán kipróbálni a jelenséget, sem pedig a felfedezett anomália formális vizsgálatára. Ennek elvégzésére viszont szeretnénk a közeli jövőben sort keríteni.

Referált folyóiratban megjelent cikkek

[1] Dávid, Á., Kozma, L., Pozsgai, T. (2008), *„On the model checking of a system consisting of many similar components”*, Annales Univ. Sci. Budapest, Sect. Comp. 28, pp. 183-195.

[2] Dávid, Á., Pozsgai, T., Kozma, L. (2007), *„Extending a system with verified components”*, Periodica Polytechnica, Vol 51, Issue 3-4, pp. 133-139.

[3] Dávid, Á., Kozma, L., Varga, L. (2006), *„On the correctness of data type classes based on contracts”*, Pure Mathematics and Applications, Volume 17, Issue 3-4, pp. 251-261, ISSN 1788-800X.

Referált konferenciakiadványban közölt teljes cikkek

[4] Dávid, Á., Kozma, L. (2009), *„Educational aspects of incremental model checking”*, Proceedings of the 3rd International Multi-Conference on Society, Cybernetics and Informatics, Vol 2, pp. 190-194, Jul 10-13, 2009, Orlando, Florida, USA, ISBN-10: 1-934272-73-6, ISBN-13: 978-1-934272-73-2.

[5] Dávid, Á., Pozsgai, T., Kozma, L. (2007), *„On the granularity of components”*, 7th International Conference on Applied Informatics, pp. 219-228, Jan 28-31, 2007, Eger, Hungary.

Konferenciakiadványban közölt teljes szövegű cikkek

[6] Dávid Á., Csatári J., Beleznay P. (2012), *„Hálózati protokollok formális szemmel”*, Networkshop 2012 (konferenciakiadvány), Veszprém, 2012. április 10-13., ISBN: 978-963-88335-1-8.

[7] Dávid, Á., Kozma, L. (2011), *„Hálózatbiztonság formális szemmel”*, Informatika a felsőoktatásban (konferenciakiadvány), 608-612. o., 2011. augusztus 24-26., Debrecen, ISBN 978-963-473-461-1.

[8] Dávid, Á. (2008), *„Formális szoftververifikációs eszközök alkalmazásának nehézségei a gyakorlatban”*, Informatika a felsőoktatásban (konferenciakiadvány), 2008. augusztus 27-29., Debrecen.

[9] Dávid, Á., Gál, B. (2008), *„A modellellenőrzés kihívásai”*, Informatika Korszerű Technikái Konferencia (konferenciakiadvány), 2008. március 7-8., Dunaújváros.

[10] Dávid, Á. (2007), *„Inkrementális modellellenőrzés a gyakorlatban”*, Felsőoktatási matematika-, fizika- és számítástechnika oktatók XXXI. Konferenciája (konferenciakiadvány), 2007. augusztus 23-25., Dunaújváros.

[11] Dávid, Á., Pozsgai, T., Kozma, L. (2005), *„Environment for concurrent programming”*, Proceedings of the 5th International Conference of PhD Students, Aug 14-20, 2005, Miskolc, Hungary.

[12] Dávid, Á., Pozsgai, T., Kozma, L. (2005), *„Educational framework for developing applications built from verified components”*, Proceedings of the Ninth Symposium on Programming Languages and Software Tools, pp. 7-18, Aug 13-14, 2005, Tartu, Estonia, published by Tartu University Press.

Irodalomjegyzék

[13] Amla, N., Du, X., Kuehlmann, A., Kurshan, R. P., McMillan, K. L. (2005), *„An Analysis of SAT-Based Model Checking Techniques in an Industrial Environment”*, LNCS 3725, pp. 254-268.

[14] Andrews, G. R. (1989), *„A Method for Solving Synchronization Problems”*, Science of Computer Programming, 13 1989/90, pp. 1-21.

[15] Bensalem, S., Bouajjani, A., Loiseaux, L., Sifakis, J. (1993), *„Property preserving simulations”*, LNCS 663, pp. 260-273.

[16] Berard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., Schnoebelen, P. (2001), *Systems and Software Verification: Model-Checking Techniques and Tools*, Springer.

[17] Biere, A., Cimatti, A., Clarke, E. M., Strichman, O., Zhu, Y. (2003), *„Bounded Model Checking"*, Advances in Computers, Vol. 58, Academic Press (pre-print).

[18] Cavada, R., Cimatti, A., Olivetti, E., Keighren, G., Pistore, M., Roveri, M., Semprini, S., Tchaltsev, A.: *NuSMV 2.5 User Manual*, http://nusmv.fbk.eu/NuSMV/userman/v25/nusmv.pdf

[19] Clarke, E. M., Grumberg, O., Long, D. E. (1994), *„Model checking and abstraction"*, ACM Transactions on Programming Languages and Systems (TOPLAS), Vol. 16, Issue 5, pp. 1512-1542.

[20] Clarke, E., Grumberg, O., Peled, D. (2000), *Model Checking*, MIT Press.

[21] Crnkovic, I., Hnich, B., Jonsson, T., Kiziltan, Z. (2002), *„Specification, Implementation and Deployment of Components"*, Communications of the ACM, Vol. 45, pp. 35-40.

[22] Dams, D., Gerth, R., Grumberg, O. (1993), *„Generation of reduced models for checking fragments of CTL"*, Proceedings of the 5th Conference on Computer-Aided Verification, LNCS 697, pp. 479-490.

[23] Dams, D., Gerth, R., Grumberg, O. (1997), *„Abstract interpretation of reactive systems"*, ACM Transactions on Programming Languages and Systems (TOPLAS), Vol. 19, Issue 2, pp. 253-291.

[24] Emerson, E. A., Sistla, A. P. (1996), *„Symmetry and model checking"*, Formal Methods in System Design – Special issue on symmetry in automatic verification, Vol. 9, Issue 1-2, pp. 105-131.

[25] Godefroid, P., Pirottin, D. (1993), *„Refining dependencies improves partial-order verification methods"*, Proceedings of the 5th Conference on Computer-Aided Verification, LNCS 697, pp. 438-449.

[26] Hopkins, J. (2000), *„Component primer"*, Communications of the ACM, Vol. 43, pp. 27-30.

[27] Huber, P., Jensen, A., Jepsen, L., Jensen, K. (1985), *„Towards reachability trees for high-level Petri nets"*, LNCS 188, pp. 215-233.

[28] Ip, C. W., Dill, D. L. (1996), *„Better verification through symmetry"*, Formal Methods in System Design – Special issue on symmetry in automatic verification, Vol. 9, Issue 1-2, pp. 41-75.

[29] Johnson, R. E. (1997) *„Frameworks = (Components + patterns)"*, Communications of the ACM, Vol. 40.

[30] Kurshan, R. P. (1990), *„Analysis of discrete event coordination"*, REX workshop Proceedings on Stepwise refinement of distributed systems: models, formalisms, correctness, pp. 414-453.

[31] Lee, S. C., Shirani, A. I. (2004), *„A component based methodology for web application development"*, Journal of Systems and Software, Volume 71, Issue 1-2, pp. 177-187.

[32] McMillan, K. L. (1993), *Symbolic Model Checking: An Approach to the State Explosion Problem*, Kluwer Academic.

[33] McMillan, K. L. (2003), *„Methods for exploiting SAT solvers in unbounded model checking"*, MEMOCODE'03 Proceedings of the First ACM and IEEE International Conference on Formal Methods and Models for Co-Design, pp. 135-144.

[34] Miller, A., Donaldson, A., Calder, M. (2006), *„Symmetry in temporal logic model checking"*, ACM Computing Surveys (CSUR), Vol. 38, Issue 3, Art. 8.

[35] Müller-Olm M., Schmidt, D. A., Steffen, B. (1999), *„Model checking: a tutorial introduction"*, Proc. of the 6th Static Analysis Symposium, G. File and A. Cortesi (eds.), LNCS 1694, pp. 330–354.

[36] Peled, D. (1994), *„Combining partial order reductions with on-the-fly model checking"*, LNCS 818, pp. 377-390.

[37] Sparling M. (2000), *„Lessons learned through six years of component-based development"*, Communications of the ACM, pp. 47-53, Vol. 43.

[38] Szyperski, C. (2002), *Component Software – Beyond Object-Oriented Programming (Second Edition)*, Addison-Wesley/ACM Press.

[39] Thang N. T., Katayama T. (2004), *„Open Incremental Model Checking (Extended Abstract)"*, Proceedings of ACM SIGSOFT Symposium on the Foundations of Software Engineering, pp. 134-137, Oct 31-Nov 1, 2004, Newport Beach, California, USA.

[40] Thang, N. T., Katayama, T. (2005), *„Specification and verification of intercomponent constraints in CTL"*, ACM SIGSOFT Software Engineering Notes, pp. 1-8, Vol. 31, Issue 2.

[41] Valmari, A. (1990), *„A stubborn attack on state space explosion"*, Proceedings of the 2nd Workshop on Computer-Aided Verification, LNCS 531, pp. 156-165.

[42] Vitharana, P. (2003), *„Risks and Challenges of Component-based Software Development"*, Communications of the ACM, Vol. 46, pp. 67-72.

[43] Yang, J (2003), *„Web Service Componentization"*, Communications of the ACM, Vol. 46, pp. 35-40.